AMENDMENTS TO THE SPECIFICATION:

Please amend the specification as follows:

On page 6, please amend the paragraph starting on line 4 as follows:

Fig. 6 is a flow diagram of processes consistent with the present invention for converting database protocol commands to ~~a~~ general computer programming language commands;

On page 11, please amend the paragraph starting on line 14 as follows:

The EJB objects 130 have methods and properties defined for accessing the appropriate tables and fields of the database. However, the client application does not have direct access to the database, but instead has access through the EJB objects. As a consequence, a map is created to correlate queries or commands from the client application to EJB objects for executing those commands. In order to map SQL protocol commands to EJB ~~objectss~~ objects to access the database, the properties and methods of the EJB objects that are used to access the database are determined to create the database bridge 120.

On page 21, please amend the paragraph starting on line 3 as follows:

After all methods are identified, design patterns are applied to identify public methods (step 910). The design patterns are typically standard design patterns that may be used to identify public methods, or a sub-set of methods which may be exported, that are associated with the class that is being analyzed. The public methods are then identified (step 912). Once the public methods are identified (step ~~914~~ 912), it

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com

is determined whether there is a base class, or another class to be analyzed (step 914). When it is determined that the top-level base class has already been analyzed, the process of identifying methods is completed (step 909). If it is determined that there is a base class (step 914), the base class is obtained to be analyzed (step 902).

On page 23, please amend the paragraph starting on line 13 as follows:

If the method "I" does have the form "get<string>" (step 1104), a search is made for a method named "set<string>" (step 1106), where "<string>" is the same in both "get<string>" and "set<string>." It should be appreciated that any suitable algorithm may be employed to search for a method named "set<string>," which may be located in the same class as the method named "get<string>." It is determined whether a method named "set<string>" has been found (step 1108). If a method named "set <string>" has not been found, "I" is incremented (step 1102). It should be understood that when only a method named "get<string>" has been found, the property identified as "<string>" may be a read-only property. Alternatively, if a method named "set<string>" has been found, a determination is made as to whether the design pattern for a simple read-write property has been met (step 1110). Although the design pattern for a simple read-write property may take on any suitable form, the design pattern is such that "set<string>" returns a void and has one argument, while "get<string>" returns a result, which is of the same type as the argument to "set<string>," and has no arguments.

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com

On page 24, please amend the paragraph starting on line 4 as follows:

If the design pattern for the simple read-write property has not been met, process flow returns to the counter where "I" is incremented (step 1102). If the design pattern for the simple read-write property has been met, "<string>" is added to a list of all simple read-write properties found (step 1112). After "<string>" is added to the list of all simple read-write properties found, process flow returns to the counter where "I" is incremented (step 1102). Steps 1102 through 1112 are repeated until no more methods remain to be checked to determine if method "I" has the form "get<string>." When no more methods remain to be checked, then the process of finding simple read-write properties ends (step 1103).

On page 25, please amend the paragraph starting on line 3 as follows:

If the method "I" has the form "is<string>" (step 1204), a search is made for a method named "set<string>," where "<string>" is the same in both "is<string>" and "set<string>**[[.]]**" (step 1206). The process determines whether a method named "set<string>" has been found (step 1208). If a method named "set<string>" is not found, process flow returns to step 1202 where "I" is incremented. However, if a method named "set<string>" is found, it is determined whether the design pattern for a boolean property is met (step 1210). It should be appreciated that the design pattern for a boolean property may take on any suitable form. For example, the design pattern may be such that "set<string>" returns a void and has one boolean argument, while "is<string>" returns a boolean, and has no arguments.

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com

On page 26, please amend the paragraph starting on line 11 as follows:

If the method "I" does indeed have the form "get<string>" (step 1304), a search is made for a method named "set~~<string~~<string>" (step 1306), where "~~string~~<string>" is the same in both "get<string>" and "set<string>." It should be appreciated that known processes may be employed to search for a method named "set<string>," which may be located in the same class as the method named "get<string>." The process determines whether a method named "set<string>" has been found (step 1308). If a method named "set<string>" has not been found (step 1308), process flow returns to step 1302 where "I" is incremented. Alternatively, if a method named "set<string>" has been found (step 1308), the process determines whether the design pattern for indexed properties are met (step 1310). Although the design pattern for indexed properties may take on any suitable form, the design pattern is such that "get<string>" returns a result and takes one integer argument, while "set<string>" returns a void, and takes two arguments, the first being an integer, and the second being of the same type as the result returned by "get<string>."

On page 27, please amend the paragraph starting on line 10 as follows:

After methods and properties associated with a class are identified, the events associated with the class are then identified. Fig. 14 is a process flow diagram that illustrates the steps involved with identifying events associated with a class in accordance with an embodiment of the present invention. Events typically provide a way for one component to notify other components that something of interest has occurred. After the class to be analyzed is obtained (step 1402), if an associated

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com

component information class ~~is in~~ exists (step 1404), the information class is queried about events contained within the information class (step 1406). If it is determined that all events are known to the information class (step 1407), the process of identifying events associated with a class ends (step 1409). If it is determined that all events are not known to the information class, all methods associated with the class that is being analyzed are found (step 1408). While any appropriate process may be used to find methods, a reflection process is typically used to identify all events associated with a class.

On page 28, please amend the paragraph starting on line 1 as follows:

After all methods are identified, design patterns are applied to identify public events (step 1410). The design patterns may be standard design patterns that may be used to identify public events that are associated with the class that is being analyzed. A public event is an event that is accessible to, *e.g.,* may be exported to, classes other than the class with which the public event is associated. One suitable process of identifying public events will be described in more detail below with respect to Fig. 15. After the public events are identified (step 1412), the process determines whether there is a base class (step 1414). When it is determined that the top-level base class has already been analyzed, the process of identifying events ends (step 1409). If a base class exists (step 1414), the base class becomes the class to be analyzed, and the process flow returns to step 1402.

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com

On page 28, please amend the paragraph starting on line 19 as follows:

If the method "I" does have the form "add<string>listener" (step 1504), a search is made for a method named "remove<string>listener" (step 1506), where "<string>" is the same in both "add<string>listener" and "remove<string>listener." It should be appreciated that known processes may be employed to search for a method named "add<string>listener." In ~~a~~ step ~~908~~1508, it is determined whether a method named "remove<string>listener" has been found. If a method named "remove<string>listener" has not been found (step ~~1502~~1508), process flow returns to step 1502 where "I" is incremented. Alternatively, if a method named "remove<string>listener" has been found (step ~~1502~~1508), the process determines whether the design pattern for a public event is met (step 1510). Although the design pattern for a public event may take on any suitable form, the design pattern is such that "add<string>listener" returns a void and has one argument, and "remove<string>listener" returns a void and has one argument which is of the same type as the argument to "add<string>listener."

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com